Difference Engine Work for NCI Protégé

January 3, 2011

1 Architecture

1.1 Difference Engine

The framework for the difference engine has already been completed. The difference between two ontologies is calculated in two phases, both of which are pluggable.

1.1.1 Calculation of Created, Deleted and Refactored Entities

This initial phase of the difference calculation examines the two ontologies to align the entities in the two ontologies. It runs a series of plugins that attempt to determine which entities in the source ontology correspond to which entities in the target ontology. As the plugins discover mappings between source entities and target entities, a constant time algorithm discovers what OWL axioms in the source and target ontologies match. These mappings of the matching entities and axioms are then sent to each of the plugins to allow them to update their calculation of the matching entities. This algorithm continues recursively until all the plugins give up.

In the case of NCI, this algorithm is overkill and can be simplified if it becomes a bottleneck (this would be an undesirable outcome). In the NCI case, OWL entites are neither deleted or renamed. So the correspondence of the OWL axioms can be determined though a set difference to determine what axioms have been added and what axioms have been removed. Similarly the created entities can be calculated as a set difference of the entities in the two ontologies.

In the case of NCI only a few generic plugins are needed to complete this stage of the calculation. One of these matches entities in the two ontologies by code. This plugin calculates the code annotation of all the entities in the two ontologies and matches the entities in the two ontologies that have the same code. The second plugin matches entities in the two ontologies that have the same id. This plugin is needed to match entities such as rdfs:label that are used in the two OWL ontologies but which cannot be matched by code.

1.1.2 Calculation of User friendly differences

At this point a collection of differences has been calculated but the differences are not expressed in a particularly easy format for a user to understand. The next portion of the computation prepares the changes for a user. The first thing that is done is that the above changes are reorganized into a map from changed entities to the changes made to those entities. Already this is a significant improvement in the presentation. Then a series of plugins analyzes these changes to try to find a representation that will be most easily understood by a user. The NCI specific plugins that we will use for this will be:

- a merge detection plugin that will use the Merge_Into annotation to detect a merge operation.
- a split detection plugin that will use the Split_From annotation property to detect a split operation.
- a retire detection plugin that will detect when a class has been moved to the retirement hierarchy.

In addition we will need some non-NCI specific plugins that will detect

- changes to an annotation property (as opposed to an annotation property value being removed and a different one being added).
- changes to a definition.
- changes to a named superclass.
- changes to an anonymous superclass.

1.2 Protégé 3 Issues

It has been requested that the difference engine demonstration be based on the Protégé 3 User tab so that we can show the interaction of the difference engine and the existing changes ontology. The tab that we will demonstrate will use the Protégé 3 change ontology mechanisms to analyze the changes ontology. In NCI Protégé this is used identify which users changed which entities.

1.2.1 Enabling the Protégé Libraries and the User Tab

In order to make the Protégé libraries available in a Protégé 4 environment we will have to modify the Protégé 3 plugins to be OSGi bundles. This is a relatively straightforward task but it some care will have to be taken to ensure that there are no class loading problems.

1.2.2 Calculation of the Changes Ontology Information

The changes ontology tends to be very large and calculation involving the changes ontology run fairly slowly. Thus the analysis of the changes ontology data will take some time and this may be an important limiting factor on performance of the NCI version of the difference engine. One approach to dealing with this may be to run this calculation in parallel with the difference engine work. The two portions of the effort are orthogonal and on a multi-cpu or dual core machine this might be an effective strategy.

1.3 Preliminary Performance Results

We have done our first run of the difference engine with none of the NCI difference engine plugins installed. In this mode the plugin calculated the difference between the two ontologies as a collection of axioms added and deleted indexed by the classes that have been changed. I ran the comparison on the thesaurus version 09.12d and 10.04f. It found 2984 created classes and 77641 modified classes. Split between these classes where 330228 axioms that were added 282865 Axioms removed. It calculated the difference between the two ontologies in just under three minutes. This time interval breaks down as:

- 28 seconds to parse the first ontology. This time probably will not be improved.
- 31 seconds to parse the second ontology. This time probably will not be improved.
- 86 seconds to search for entity creates, deletes and renames. Since NCI doesn't do renames and deletes this part of the algorithm could be simplified for the NCI case. But the time spent in this algorithm will not change as we add plugins so it may be acceptable.
- 26 seconds to group the changed axioms by the created/modified classes. This part of the algorithm will take longer as our development work progresses.

2 Plan

Tasks:

1. Build Protégé 3 portion of the user interface. This task can start immediately and does not depend on any other tasks.
   (a) convert the protege 3 plugins to be visible to OSGi (1 1/2 weeks, started)

   (b) make a copy of the classes from the UsersTab so that they are available as a Protégé 4 plugin (probably a view) and can be modified as needed (2 weeks).

2. early demonstration of performance of the difference engine. This task will not involve any work with the graphical interfaces and can be started immediately. It will be updated as the plugins in task 3 are developed.

   (a) evaluate the overhead of processing the Protégé 3 changes ontology and determine if this overhead can be run in parallel with the difference calculation. It is quite possible that this will be the limiting factor on the performance of the NCI difference engine.
   (b) as NCI difference engine plugins are created, run the difference engine on a real copy of the baseline and final version of the NCI ontology and compare how this behaves relative to Protégé 3 Prompt. The first version of these results have been reported above. This task will be updated as the plugins in task 3 are developed.

3. build missing difference engine plugins. Conceivably this work does not need to be completed to demonstrate the concept. When plugins are missing the differences will simply not be expressed in as simple a form as they would be in the final version of the tool. This task does not depend on any other tasks and can be started immediately.

   (a) annotation change detection plugin
   (b) definition change detection plugin
   (c) merge detection plugin (NCI specific)
   (d) split detection plugin (NCI specific)
   (e) retire detection plugin (NCI specific)

4. integrate the difference engine results into the copy of the Protégé 3 user tab graphical user interface. This task cannot be started until task 1 is complete. This task cannot complete until task 3 is completed.

5. revise the current difference engine architecture as new requirements are discovered. This task will run concurrently with the other work.

6. build a plugin mechanism for difference engine plugins. This is perhaps on optional task as it is possible that we can hardwire the use of the various difference engine components for the initial demonstration.

7. investigate the ability to send changes calculated during the user interaction with the difference engine results to a Protégé 3 database backend project. This task does not depend on any of the other tasks.

8. demonstration and education about the use of the tool. This task depends on the completion of task 4.